

Санкт-Петербургский Государственный Университет
Математическое обеспечение и администрирование информационных
систем

Информационно-аналитические системы

Волжина Елена Григорьевна

Рекомендательная система для образовательного контента

Бакалаврская работа

Научный руководитель:
ст. преп. Ярыгина А. С.

Рецензент:
Вяххи Н. И.

Санкт-Петербург
2016

SAINT-PETERSBURG STATE UNIVERSITY
Software and Administration of Information Systems

Analytical Information Systems

Elena Volzhina

Recommender system for educational content

Bachelor's Thesis

Scientific supervisor:
senior assistant prof. Anna Yarygina

Reviewer:
Nikolay Vyahhi

Saint-Petersburg
2016

Оглавление

Введение	4
1. Существующие рекомендательные системы	6
2. Платформа stepic.org	9
2.1. Терминология	9
2.2. Возможности	10
3. Реализация рекомендательной системы	12
3.1. Информация о пользователе	12
3.2. Графы переходов	13
3.3. Хендлеры (способы рекомендации)	15
3.4. Оценка реакции пользователя	18
3.5. Формирование выдачи	18
3.5.1. Линейная регрессия	19
4. Адаптивная рекомендательная система	23
4.1. MathsGarden	24
4.1.1. Выбор наиболее подходящей задачи	24
4.1.2. Рейтинг Эло	25
4.1.3. Механика работы	26
4.1.4. Результаты работы	27
4.2. Адаптивность в stepic.org	28
5. Анализ результатов	30
5.1. Простые рекомендации	31
5.2. Контекстные рекомендации	32
5.3. Адаптивные рекомендации	33
6. Заключение	36
Список литературы	37

Введение

В современном мире онлайн-образование постепенно становится все более популярным. Возможность учиться у профессоров ведущих учебных заведений, изучать новые области, получать нужные в работе знания, не выходя из дома, привлекает большое количество людей.

Одной из наиболее распространенных форм онлайн-обучения являются массовые открытые онлайн-курсы (МООС, massive open online course). Чаще всего они включают видео, слайды и текстовый контент, подготовленные преподавателем, а также задачи для проверки знаний, которые обычно проверяются автоматически, но также возможна проверка студентами работ своих товарищей. В качестве задач могут быть предложены самые разнообразные типы заданий: от простого выбора правильного ответа до задач на программирование и написания эссе.

У онлайн-образования есть свои особенности, отличающие его от обычного, офлайн-образования. Среди плюсов, во-первых, уже упомянутая выше доступность каждому, у кого есть доступ к интернету. Во-вторых, это почти неограниченная масштабируемость: благодаря автоматизированной проверке задач на курсе могут одновременно учиться тысячи человек, что несопоставимо с обычными курсами в учебных аудиториях. В-третьих, каждый студент может выбирать удобное для себя время и темп для прохождения материала. В-четвертых, в распоряжении преподавателей оказывается большое число данных о том, как пользователи проходят его курсы, которые он может использовать для анализа и улучшения своих материалов.

В то же время в онлайн-обучении есть и минусы. В отличие от традиционного образования, где у студента всегда есть мотивация в виде оценки его академической успеваемости, в случае онлайн-курсов нет никаких штрафов за не пройденный курс. Из-за этого доля закончивших курс из записавшихся на него редко превышает 10%. Помимо этого, из-за большого числа учащихся у преподавателя нет никакой возможности уделять индивидуальное внимание каждому

Таблица 1: Платформы с онлайн-курсами

Название	Год запуска	Пользователей
Coursera[2]	2012	15 млн
edX[29]	2012	5 млн
Udacity[26]	2012	1.6 млн
Stepic.org[22]	2013	180 тыс.

студенту сообразно его уровню и возможностям.

В таблице 1 представлены сведения о нескольких популярных платформах для онлайн-курсов. Последняя из них, *stepic.org*, содержит в основном материалы на русском языке. В рамках этой работы была разработана и исследована рекомендательная система для этого сайта.

Задача этой работы – создать рекомендательную систему, которая могла бы посоветовать студенту контент, который будет интересен ему, и которая также будет учитывать уровень подготовки студента, его знания и пробелы. Кроме этого система должна уметь оценивать сложность контента. Это нужно, в частности, для адаптивных рекомендаций, которые будут помогать пользователю изучать материал, гибко подстраиваясь под него, предлагая именно тот контент, который ему нужен сейчас для обучения.

Такая система будет полезна пользователям персонализированными рекомендациями, которые могут помочь им изучить конкретную тему или предложить что-то новое.

1. Существующие рекомендательные системы

Тема рекомендательных систем активно исследуется последние десятилетия[27][13][1]. Она широко применима на практике, в том числе в коммерции, что значительно стимулирует ее развитие, как в плане теоретических исследований, так и в виде практических задач.

В качестве одного из первых примеров рекомендательной системы в современном представлении можно привести *movielens.org*[15], предлагающий пользователям фильмы на основе их предпочтений. Этот сервис интересен тем, что он предоставляет всем желающим обширный набор данных о фильмах и рейтингах, поставленных им пользователями. Этот набор данных был использован в большой числе исследований в области рекомендательных систем последних двух десятилетий.

Существует два основных класса рекомендательных систем[19]:

- Системы, основанные на *фильтрации контента*. Такие системы предлагают пользователям контент, похожий на тот, что они изучали ранее. Схожесть подсчитывается с помощью характеристик сравниваемых объектов. Например, для рекомендации фильмов можно использовать близость жанров или актерский состав. Такой подход используется в сервисе для оценки, поиска и рекомендаций фильмов *Internet Movie Database*[9].
- Системы, использующие *коллаборативную фильтрацию*. В этом случае пользователю предлагается контент, заинтересовавший похожих на него пользователей. Рекомендации сервиса MovieLens основаны именно на этом подходе.
- *Гибридные* системы, комбинирующие два предыдущих подхода. Система такого типа используется в *Netflix*[16], сервисе для онлайн-просмотра фильмов и сериалов.

В данной работе будет создана гибридная система с более активным использованием фильтрации контента и менее активным – коллаборативной фильтрации.

Существует множество исследований, посвященных рекомендательным системам для обучения, основанного на использовании технологий (*Technology Enhanced Learning*)[30]. Специфика задачи в этом случае добавляет новые направления развития рекомендательной системы. Во-первых, это возможность построения адаптивной рекомендательной системы, которая будет подстраиваться под нужды пользователя в конкретный момент и предлагать ему оптимальные пути изучения материала[23]. В таком формате могут быть реализованы различные тренажеры, например, по математике или какому-либо языку программирования, содержащие набор задач разной сложности, из которых разным ученикам будут в каждый момент времени подходить разные. Во-вторых, можно извлечь зависимости между обучающими материалами из данных о том, как пользователи их проходят. Эти данные могут помочь выделить отдельные темы в материалах, связи между этими темами, их соотношения по сложности[7].

Упомянутые ранее платформы для онлайн-обучения используют свои рекомендательные системы, советуя пользователям курсы, которые их могут заинтересовать. Недостаток этих рекомендаций в том, что они могут предложить только курс целиком, но не какую-то его часть, даже если пользователю будет интересна только она. Также построенная таким образом система никак не может помочь пользователю в изучении курса, который он выбрал.

Рекомендательная система ресурса MathsGarden[10] работает, напротив, с самыми небольшими частями контента – отдельными задачами. Она представляет собой тренажер по элементарной арифметике для учеников начальной школы, который предлагает ученику задачи, оптимально подходящие ему в данный момент времени по сложности. Для этого система подсчитывает и динамически изменяет относительную характеристику знаний ученика, а также

характеристику сложности задач. Эта система будет подробно рассмотрена позже.

2. Платформа stepic.org

Платформа *stepic.org*[22] – сайт с большой русскоязычной аудиторией, на котором размещено несколько десятков онлайн-курсов, преимущественно технической тематики. Каждый из них прошли от нескольких сотен до нескольких тысяч человек. Ежедневно на платформу заходят тысячи учащихся. Это крупнейший в России ресурс для онлайн-образования.

2.1. Терминология

Курсы на stepic.org разделены на *модули*, каждый модуль обычно рассчитан на неделю или две. У модуля могут быть установлены дедлайны на решение задач. После жесткого дедлайна пользователь все еще может проходить модуль, но уже не получит баллов за решение задач. Еще у модуля может быть мягкий дедлайн, после которого пользователь получит за решение задач часть баллов.

Модуль состоит из *уроков*, каждый из которых посвящен отдельной небольшой теме. Обычно уроки формируются так, чтобы можно было рассматривать урок в отрыве от курса, то есть как отдельную структурную единицу.

На рис. 1 можно увидеть пример структуры курса, состоящего из трех модулей.

В каждом уроке есть несколько *шагов* (*стэпов*), представляющих собой либо лекционный материал в виде видеозаписи или текста, либо задачу одного из множества видов. Бывают задачи-тесты с выбором ответа, задачи с текстовым ответом, задачи на программирование и многие другие. Большинство из них предполагают моментальную автоматическую проверку системой, но есть также задачи на написание эссе, которые студенты сами проверяют друг у друга на основе сформулированных преподавателем критериев.

Каждый урок или курс может быть помечен *тегом* – концепцией, которая описывает его содержимое. Примеры таких тегов: программирование, linux, статистика. Каждый тег привязан к объекту

























1. Вводный модуль		0 / 0
	1.1 Урок 1  Урок от: Lena Volzhina  0  0	▼
	1.2 Урок 2  Урок от: Lena Volzhina  0  0	▼
2. Основной модуль		0 / 11
	2.1 Урок 1  Урок от: Lena Volzhina  0  0 — 0 / 4	▼
	2.2 Урок 2  Урок от: Lena Volzhina  0  0 — 0 / 3	▼
	2.3 Урок 3  Урок от: Lena Volzhina  0  0 — 0 / 4	▼
3. Заключительный модуль		0 / 0
	3.1 Урок 1  Урок от: Lena Volzhina  0  0	▼

Рис. 1: Структура курса

в базе знаний Wikidata[28].

Уроки могут быть объединены в *пути*. В отличие от курсов, у путей нет модулей, дедлайнов и возможности получить сертификат за их прохождение. Это просто способ объединять уроки в логическую последовательность. Обычно после того, как курс закрывается для просмотра, его уроки собираются в путь и выкладываются в библиотеку.

2.2. Возможности

В основном студенты приходят на stepic.org, чтобы изучать курсы. Успешно прошедшие курс и набравшие нужное количество баллов получают электронные сертификаты. Преподаватели могут установить границу для получения обычного сертификата, и для получения сертификата с отличием.

Помимо курсов на stepic.org также есть раздел уроков. Уроки, которые были включены в один курс, здесь располагаются без связи

друг с другом.

Stepic.org позволяет пользователям создавать свои уроки и курсы, давать ссылки на них или же встраивать в другие сайты. Также имеется возможность интеграции с любой платформой через LTI, благодаря чему в некоторых курсах на платформе Coursera[2] используются задачи, которые создаются и проверяются на stepic.org. Весь контент на платформе распространяется под открытой лицензией Creative Commons[3].

Пользователи могут общаться между собой с помощью комментариев под каждым шагом. Там обычно обсуждают лекционный материал, помогают в решении задач или общаются с преподавателями.

Для удобства обучения на платформе разработана рекомендательная система, описанная в этой работе. Она помогает пользователям находить интересные им материалы.

3. Реализация рекомендательной системы

В рамках данной работы была реализована рекомендательная система, которая предлагает пользователю новые уроки на основании того, чем он интересовался ранее. Она использует для этого различные способы найти контент, чем-то похожий на тот, который пользователь уже изучал, или же не столь похожий, но в целом интересный. Каждый из таких способов оформлен в виде отдельной функции (*хендлера*), которая извлекает список уроков по заложенному в нее принципу и размечает их весами в зависимости от того, насколько они подходят пользователю. Подробное описание этих функций-хендлеров будет представлено ниже.

Рекомендательная система может применяться в разных ситуациях. Два основных случая:

- просто рекомендация для конкретного пользователя, которая показывается ему на главной странице и во вкладке «Рекомендованные уроки», а также время от времени присылается на почту;
- *контекстная* рекомендация, которая показывается пользователю после прохождения урока вне курса (то есть урока, для которого нет фиксированного следующего за ним), как подсказка, что посмотреть дальше.

Вместе с рекомендацией пользователю также показывается информация о том, почему ему был предложен именно этот урок. Эту информацию легко извлечь благодаря тому, что в каждую функцию-хендлер заложена какая-то естественная интуиция, так что можно восстановить причину рекомендации.

3.1. Информация о пользователе

В процессе обучения на stepic.org пользователь оставляет значительный цифровой след. Помимо информации о его прогрессе

сохраняются также сведения о том, когда он посещал разные шаги, когда пытался решать задачи и с каким результатом, и даже как он смотрел видео: какие установил скорость и громкость, где останавливался и где перематывал.

Наиболее интересны для базовой рекомендательной системы следующие сведения: теги, которыми пользователь интересовался, уроки, на которые он заходил, уроки, которые он прошел, а также, если рекомендация контекстная, текущий урок.

Кроме информации о конкретном пользователе, сохраненные данные можно рассматривать как сведения о взаимодействии пользователей с контентом. Одним из способов агрегировать и использовать эти сведения являются графы переходов, о которых подробно рассказано в следующем параграфе.

3.2. Графы переходов

Часто бывает полезно узнать, как пользователи изучают контент, что они смотрят чаще, что реже, какие материалы проскакивают, на какие возвращаются, и, что немаловажно, в каком порядке их смотрят. Используя сохраненные данные о действиях пользователей, несложно дать ответы на эти вопросы. Для этого был использован *граф переходов* (*process map*), то есть такой ориентированный граф, в котором вершинами будут единицы контента (например, шаги или уроки), а веса дуг будут показывать количество переходов между этими единицами контента.

Многим действиям, которые совершают пользователи на платформе, соответствуют записи в таблице *событий*. У каждого события есть временная метка, что позволяет расположить события на временной шкале. События разделены на типы, соответствующие действиям пользователя, например, «viewed» (открыл шаг), «succeeded attempt» (решил задачу) или «commented» (оставил комментарий). Нас будут интересовать действия пользователя по отношению к шагам, которые затем легко обобщить на уроки, к которым эти шаги

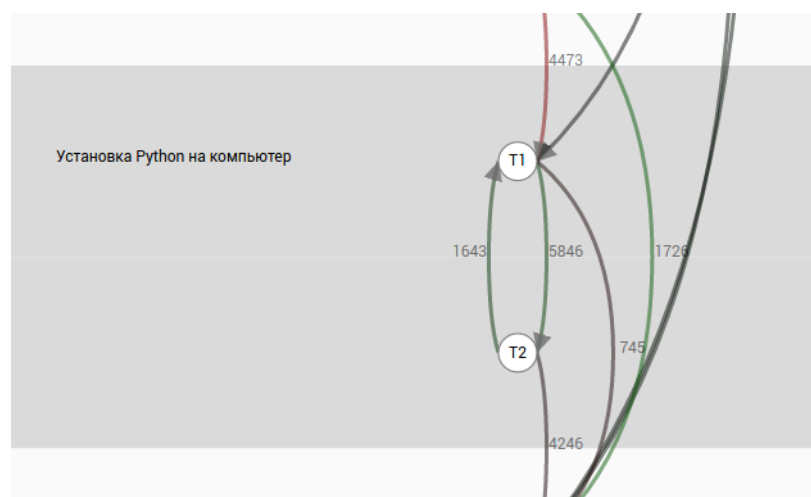


Рис. 2: Граф переходов для шагов одного урока

принадлежат, и даже на теги, которыми уроки помечены.

Визуально граф переходов можно представить себе как ряд вершин-материалов (например, шагов), между некоторыми из которых есть дуги. Наличие дуги между двумя шагами означает, что существуют пользователи, которые просмотрели их друг за другом. Каждая дуга обладает меткой, содержащей число таких пользователей. Пример графа переходов для шагов одного урока внутри курса можно увидеть на рис. 2.

Процесс построения графа переходов тривиален: для каждого пользователя мы рассматриваем все его действия по отношению к шагам, сортируем их по времени, и затем подсчитываем для каждой пары шагов, сколько раз он совершал действие с одним из них непосредственно после действия с другим. Далее эти частоты суммируются для всех пользователей, и получается общий граф переходов.

Изначально графы переходов по шагам использовались, чтобы преподаватели могли изучить статистику по своим курсам, посмотреть, в каком порядке пользователи смотрят материалы, где есть странные переходы. Для простоты анализа по курсу вычислялась средняя частота перехода между шагами (с учетом того, теория в шаге или задача, а также с учетом положения шага в уроке), и ребра в итоговом графе окрашивались зеленым, если переход по ним происходил чаще

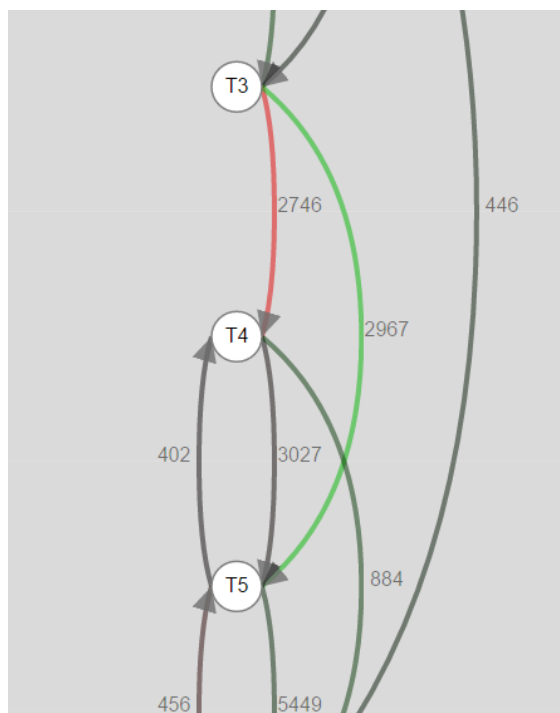


Рис. 3: Аномалия в графе переходов

ожидаемого, или красным, если реже. Фрагмент графа переходов по одному из курсов представлен на рис. 3. По этому фрагменту видно, что шаг с теорией Т4 чаще обычного пропускают и переходят к следующему за ним.

Далее графы переходов стали строиться не только по шагам, но также по урокам и тегам, чтобы использовать это в рекомендательной системе. Граф переходов по урокам строится по тому же принципу, что и для шагов, просто действия над шагами рассматриваются как действия над уроками, к которым эти шаги принадлежат. Для тегов процесс аналогичен: действия со шагами переводятся в действия с тегами их уроков.

3.3. Хендлеры (способы рекомендации)

В этом параграфе будут перечислены функции-хендлеры, каждая из которых реализует свой способ рекомендаций. Хендлер принимает в качестве параметров пользователя, а также урок, на котором тот находится, если рекомендация контекстная. Хендлер возвращает

для пользователя j список уроков, в котором каждому уроку i соответствует вес $weight_{ij} \in [0, 1]$, чем он больше, тем лучше рекомендация: $[(i, weight_{ij})]$.

Пусть мы хотим порекомендовать пользователю j уроки по конкретному тегу, для которого доля пройденных пользователем уроков составляет $tag_progress_j$. В случае, если по нему нет путей, все его уроки получают вес $tw_{ij} = tag_progress_j$. Если же путь есть, то для каждого урока этот вес делится на расстояние до ближайшего предшествующего в пути урока, пройденного пользователем j : $tw_{ij} = tag_progress_j / dist_{ij}$. Таким образом реализуется идея, что пользователю может быть интересно проходить материалы, расположенные в пути недалеко от уже изученных им.

- *Уроки с интересными пользователю тегами* – хендлер советует уроки, помеченные тегами, которые пользователь уже изучал. Для этого используется описанная выше рекомендация по тегам, вес урока i для пользователя j будет составлять $weight_{ij} = tw_{ij}$.
- *Незаконченные уроки* – рекомендуются уроки, которые пользователь начинал и не закончил, с весом тем большим, чем большую часть урока пользователь прошел: $weight_{ij} = lesson_progress_{ij}$, доля урока i , пройденного пользователем j .
- *Популярные уроки* – хендлер не использует информацию о пользователе, а советует просто самые популярные за последнюю неделю уроки на платформе. Вес прямо пропорционален популярности: $weight_{ij} = 1/n_i$, где n_i – номер урока i в списке популярных уроков.
- *Пути по урокам* – используются пути, содержащие уроки, которые пользователь проходил, рекомендуются уроки следом за пройденными, с весом тем меньшим, чем дальше урок от уже пройденных: $weight_{ij} = 1/dist_{ij}$, где $dist_{ij}$ – расстояние от урока i до ближайшего предшествующего в пути урока, пройденного пользователем j .

- *Граф переходов по тегам* – рекомендуются уроки с тегами, которые изучают после тегов, интересующих пользователя. Веса зависят от прогрессов пользователя по тегам, а также от относительных частот переходов между тегами. Пусть $freq$ – частота перехода от тега, уроки которого пользователь изучал, к некоторому тегу t . Тогда для урока i , помеченного тегом t , вес рекомендации пользователю j будет $weight_{ij} = freq \cdot tw_{ij}$.
- *Уроки по тегам от похожих пользователей* – на основе интересующих пользователя тегов выявляются пользователи с похожими на его интересами, и рекомендуются уроки с тегами, которые они сами изучают. Пусть у текущего пользователя j нашелся похожий на него пользователь k , причем мера схожести между ними – s (лежит между 0 и 1, чем она больше, тем более схожи пользователи). Тогда урок i , помеченный тегом, который изучал k , будут советовать с весом $weight_{ij} = s \cdot tw_{ij}$. Этот хендлер реализует способ рекомендаций с помощью колаборативной фильтрации.
- *Граф переходов по урокам* (только при контекстных рекомендациях) – хендлер советует уроки, следующие за текущим в графе переходов, вес зависит от относительной частоты перехода между текущим уроком и советуемым. Если переход из текущего урока в урок i совершается с относительной частотой $freq_i$, то $weight_{ij} = freq_i$.
- *Пути по урокам* (только при контекстных рекомендациях) – советуются уроки, следующие за текущим в каких-либо путях, веса обратно пропорциональны расстоянию между уроками: $weight_{ij} = 1/dist_i$, где $dist_i$ – расстояние от урока i до текущего урока.
- Если все предыдущие хендлеры в совокупности посоветовали меньше уроков, чем было нужно, в рекомендации добавляются случайные уроки, вес таких уроков равняется 0.

3.4. Оценка реакции пользователя

Чтобы понять, насколько успешна оказалась рекомендация, после ее показа фиксируется, перешел ли пользователь по ссылке, а также какую часть от урока пользователь прошел. В то же время у пользователя есть возможность узнать, почему урок был посоветован (варианты соответствуют хендлерам) и пометить рекомендацию как не интересную ему. Также мы располагаем сведениями о том, какую часть урока пользователь прошел (значение от 0 до 1).

В результате каждой показанной рекомендации мы можем сопоставить некоторое число: -1 соответствует отказу от рекомендаций, 1 – переходу по ссылке без прохождения урока, 2 – переходу по ссылке и полному прохождению урока, значение от 1 до 2 – неполному прохождению урока, а 0 – отсутствию реакции.

3.5. Формирование выдачи

Итак, мы получили списки уроков от нескольких хендлеров, в которых каждому уроку сопоставлен вес от 0 до 1, чем больше, тем более удачной оценивает хендлер рекомендацию. Встает вопрос, как их лучше всего комбинировать

Можно было бы просто для каждого урока сложить или перемножить веса хендлеров, которые его посоветовали. Но это означало бы, что мы полагаем разные хендлеры одинаково полезными, что, вообще говоря, может быть совсем не так. Хотелось бы оценить каждый хендлер коэффициентом полезности, используя при этом реакции пользователей на рекомендации. Таким требованиям удовлетворяет линейная регрессия.

На рис. 4 схематически изображен этот процесс. Регрессионная модель обучается с использованием накопленных данных о том, как рекомендации разных хендлеров были оценены пользователем. Исходя из этого модель формирует вектор весов для хендлеров, вес хендлера соответствует его полезности и влияет на его вклад в выдачу. Далее, когда требуется конкретная рекомендация, регрессионная

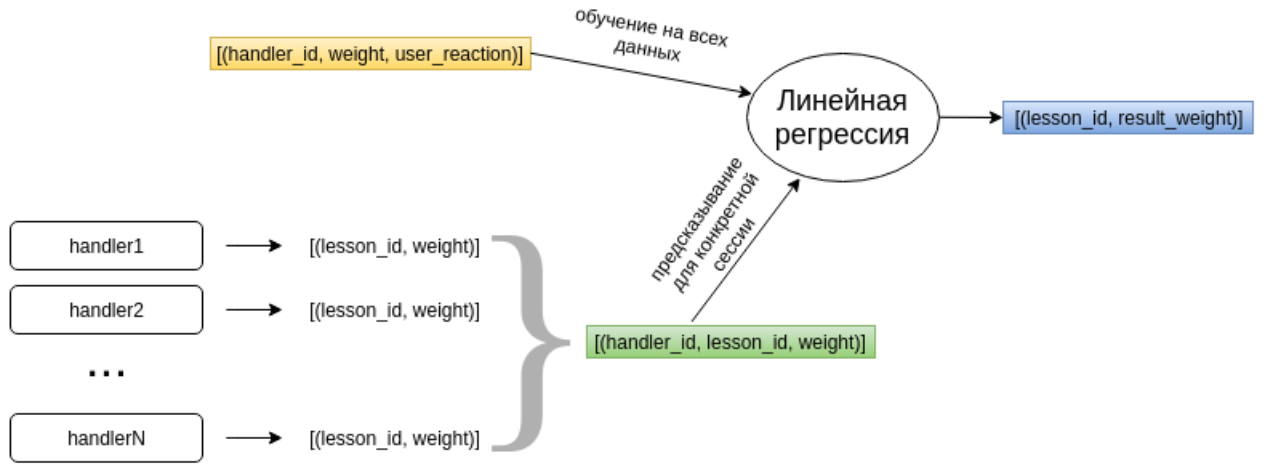


Рис. 4: Использование линейной регрессии

модель для каждого урока комбинирует веса предложивших его хендлеров с использованием своего вектора весов для них, и выдает результирующий вес для каждого рекомендованного урока.

Ниже представлено формальное описание регрессионной модели.

3.5.1. Линейная регрессия

Пусть у нас есть матрица $X = \{x_i^j\}_{i \in \{1, \dots, n\} j \in \{1, \dots, d\}}$, $x_i \in \mathbb{R}^d$, строки которой $x_i = (x_i^1, \dots, x_i^d)$ называются *наблюдениями*, а столбцы *факторами*, и столбец значений *целевой* или *зависимой* переменной $y = \{y_i\}_{i=1}^n$. Регрессионная модель $y = f(X, b) + \epsilon$, где b – параметры модели, а ϵ – случайная ошибка, называется *линейной*, если зависимость целевой переменной от факторов является линейной, то есть $f(x, b) = b_0 + \sum_{i=1}^d b_i \cdot x_i$. Обычно для удобства b_0 (*константу*) вносят под знак суммы, добавив в матрицу наблюдений столбец из единиц: $x_i^0 = 1$ для всех $i \in \{1, \dots, n\}$.

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1^0 & x_1^1 & \dots & x_1^d \\ x_2^0 & x_2^1 & \dots & x_2^d \\ \vdots & \vdots & \ddots & \vdots \\ x_n^0 & x_n^1 & \dots & x_n^d \end{pmatrix} \quad y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \quad \beta = \begin{pmatrix} b_0 \\ \vdots \\ b_d \end{pmatrix} \quad \epsilon = \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

Используя эти обозначения, линейную регрессионную модель можно выписать как $y = X\beta + \epsilon$. Решением этой задачи будем считать столбец $\hat{\beta}$, который минимизирует сумму квадратов отклонений предсказываемых значений от реальных, то есть $\hat{\beta} = \arg \min_{\beta \in \mathbb{R}^{d+1}} \sum_{i=1}^n (y_i - x_i \beta)^2$. Такой способ аппроксимации называется *методом наименьших квадратов* (*ordinary least squares*) и является наиболее широко применимым в контексте решения задачи линейной регрессии.

В нашем случае мы можем рассматривать в качестве наблюдений, то есть строк матрицы X , рекомендованные уроки, в качестве факторов этих наблюдений веса, которые хендлеры назначили уроку, а в качестве значений целевой переменной – оценку рекомендации пользователем.

Таким образом, для уроков i_1, \dots, i_n , посоветованных пользователям j_1, \dots, j_n соответственно, и для хендлеров $1, \dots, 9$ матрица наблюдений X и столбец значений целевой переменной (реакции пользователя) y будут выглядеть следующим образом:

$$X = \begin{pmatrix} weight_{i_1 j_1}^1 & weight_{i_1 j_1}^2 & \cdots & weight_{i_1 j_1}^9 \\ \vdots & \vdots & \ddots & \vdots \\ weight_{i_n j_n}^1 & weight_{i_n j_n}^2 & \cdots & weight_{i_n j_n}^9 \end{pmatrix} y = \begin{pmatrix} r_{j_1} \\ \vdots \\ r_{j_n} \end{pmatrix}$$

Вектор $\hat{\beta}$, который дает минимальную ошибку предсказания реакции пользователя, содержит индивидуальный вес для каждого из хендлеров, на который нужно домножать его рекомендации. Если вес какого-то из хендлеров относительно большой, значит, этот хендлер вносит положительный вклад в рекомендации, и наоборот.

Для работы рекомендательной системы регрессионная модель будет на регулярной основе обучаться на данных о реакции пользователей на рекомендации. Мы будем использовать библиотеку SciPy[20] для решения этой задачи и нахождения столбца $\hat{\beta}$. В этой библиотеке уже реализовано решение задачи линейной регрессии.

Регуляризация После первичной реализации рекомендательной системы с линейной регрессией, которая находила столбец весов для хендлеров $\hat{\beta}$, минимизирующий квадрат отклонений реальной реакции пользователя от предсказанной ($\hat{\beta} = \arg \min_{\beta \in \mathbb{R}^{d+1}} \sum_{i=1}^n (y_i - x_i \beta)^2$), стало заметно, что постепенно вес одного хендлера неограниченно возрастает, в то время как остальные уменьшаются, что в итоге привело к формированию выдачи практически только из результатов этого хендлера. Такой эффект называется *положительной обратной связью*, и характеризуется тем, что отклонение в результатах работы системы влияет на ее дальнейшую работу, причем чем дальше, тем больше результаты сдвигаются в сторону этого отклонения.

Помимо этого, так как хендлеры могут предлагать очень схожие рекомендации, в наших данных также может присутствовать проблема *мультиколлинеарности* факторов, что влечет за собой слабую обособленность матрицы X и, как следствие, нестабильность решения.

В результате мы получаем решение, которое дает маленькую ошибку на данных, на которых оно обучается, и большую на реальных данных. Эта ситуация называется *переобучением* (*overfitting*) модели.

В качестве решения проблемы переобучения можно рассмотреть регуляризацию. Согласно книге *The Elements of Statistical Learning*[8], основные способы регуляризации – это *лассо* (LASSO, least absolute shrinkage and selection operator [24]) и *гребневая регрессия* (регуляризация Тихонова, ridge regression [25]). Оба этих метода меняют выражение, которое мы минимизируем в процессе поиска решения регрессии, добавляя к нему штраф на норму вектора β .

В случае лассо-регуляризации используется l_1 -норма и решение находится как

$$\hat{\beta} = \arg \min_{\beta \in \mathbb{R}^{d+1}} \left(\sum_{i=1}^n (y_i - x_i \beta)^2 + \lambda \|\beta\|_1 \right).$$

В случае же гребневой регрессии используется l_2 -норма и решение

выглядит как

$$\hat{\beta} = \arg \min_{\beta \in \mathbb{R}^{d+1}} \left(\sum_{i=1}^n (y_i - x_i \beta)^2 + \lambda \|\beta\|_2 \right).$$

В обоих случаях параметр λ подбирается в процессе оптимизации.

Использование регуляризации методом лассо уменьшает все β_i , а те, что и так были относительно небольшими, становятся равны нулю. Таким образом, метод лассо хорошо подходит для выбора значащих факторов (feature selection).

Метод гребневой регрессии также уменьшает веса факторов, но при этом никогда не сводит их к нулю, если только не $\lambda = \infty$.

Согласно работе [17] лассо-регуляризация работает лучше гребневой в ситуации, когда количество факторов значительно превосходит число обучающих наблюдений. В обратной же ситуации более уместна гребневая регрессия. Соответственно, для нашего случая лучше подходит именно она. Её реализация также присутствует в библиотеке SciPy.

4. Адаптивная рекомендательная система

В рамках классического офлайн-обучения важную роль играет постоянный контроль преподавателем уровня знаний обучающегося, который позволяет своевременно скорректировать программу, дать дополнительные задачи для закрепления плохо усвоенного материала, а также подсказать ресурсы для восполнения пробелов в знаниях. В контексте массового онлайн-обучения такой возможности нет, так как студентов может быть от нескольких сотен до десятков тысяч. Вследствие этого важной характеристикой рекомендательной системы является *адаптивность*, то есть способность системы подстраиваться под уровень знаний конкретного пользователя, учитывая, что этот уровень к тому же меняется в процессе обучения.

Одной из первых задач в этой области было *компьютерное адаптивное тестирование* (*Computerized Adaptive Testing, CAT*)[31]. Суть этой задачи в том, чтобы оценивать знания в некоторой области с помощью теста, но не одного для всех учащихся, а индивидуальных, составляющихся динамически в процессе прохождения теста из имеющегося набора задач, чтобы сократить длину теста, не заставляя пользователя решать слишком простые или сложные для него задачи. Вопросы теста оцениваются различными параметрами (в основном используются сложность и дискриминативность, то есть способность задачи отделить в рамках рассматриваемой области слабых учащихся от сильных), информация о знаниях пользователя также комбинируется в некоторый общий параметр, и на каждом шаге пользователю предлагается такая задача, которая будет примерно той же сложности, какого уровня знания пользователя в данный момент, и при этом максимальной дискриминативности. Это позволяет сократить количество вопросов в тесте, не теряя при этом качество оценки. В качестве примера использования такого подхода будет рассмотрена система адаптивных тренировок по математике для учеников начальных классов MathsGarden[10].

Ещё одной важной чертой адаптивной системы является

возможность использовать связи между контентом, подсказывать пользователю, что ему изучить, чтобы лучше понимать текущий материал. В качестве примера такой системы можно привести платформу Knewton[11], которая предоставляет функциональность для изучения любого материала в адаптивном режиме. В числе прочего для этого активно используется *граф знаний*, описывающий связи и зависимости между темами материала. С использованием этого графа ученику, не справляющемуся с темой, можно подсказать материалы по теме, предшествующей текущей в графе знаний, например, для понимания алгоритма Евклида нужно сперва разобраться с тем, что такое наибольший общий делитель. Такой подход требует обширной ручной разметки материала.

4.1. MathsGarden

Сервис адаптивных математических тренировок для учащихся младшей школы MathsGarden[10] был разработан в Нидерландах в 2010 году. В качестве основы была взята обычная методика компьютерного адаптивного тестирования: из набора доступных задач выбирается такая, которая лучшим образом подходит пользователю по сложности в данный момент, результат решения анализируется, информация об уровне пользователя обновляется, процесс итеративно повторяется, пока не будет выполнен терминирующий критерий (например, ограничение по времени или требуемая уверенность в оценке студента).

Модель, использовавшаяся в этом сервисе, была использована и в нашей рекомендательной системе. Нижу я опишу эту модель подробнее.

4.1.1. Выбор наиболее подходящей задачи

Для определения подходящей пользователю задачи сегодня чаще всего используется стохастическая теория тестов (*IRT, Item Response Theory*)[31], основополагающей чертой которой является предположение, что вероятность решения пользователем задачи суть

функция от параметров пользователя и задачи. Она получила широкое распространение на фоне классической теории тестирования, так как в отличие от неё допускала различие в сложности задач. В общем случае используется логистическая функция трёх параметров (*three parameter logistic function, 3PL function*): вероятность пользователю с уровнем знаний θ решить задачу i выражается как $p_i(\theta) = c_i + \frac{1-c_i}{1+e^{-a_i(\theta-b_i)}}$, где b_i – сложность задачи, a_i – параметр дискриминативности, а c_i – шанс случайно угадать ответ на задачу [12]. При этом не обязательно использовать все параметры, в частности, создатели MathsGarden используют лишь один из них, сложность задачи, что дает однопараметрическую (1PL) модель, также известную как модель Раша[18].

Существуют математическое доказательство того, что наиболее быстро и точно провести адаптивное тестирование можно при выборе таких задач, сложность которых равняется текущей оценке знаний студента. При таком выборе средний шанс решить задачу – 50%, что в целом демотивирует студента[5]. Создатели MathsGarden, основываясь на исследовании своих коллег[14], сделали средний шанс около 75%, что ухудшило качество, но затем привнесли в оценку исхода «игры», то есть решения задачи, информацию о затраченном на решение времени. После этого качество оценки вернулось к исходным показателям[10].

4.1.2. Рейтинг Эло

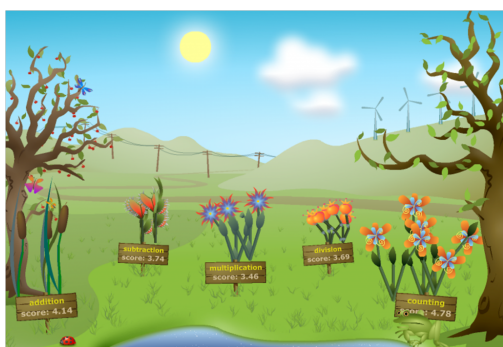
Одной из основных проблем при использовании САТ является необходимость заранее разметить задачи параметрами, использующимися в психометрической модели, определяющей наиболее подходящий вопрос. Чтобы избежать ручной разметки, исследователи совместили адаптивное тестирование с динамическим вычислением сложности задач и способностей пользователя, используя для этого модификацию системы рейтингования в шахматах, разработанную Арпадом Эло в 1978 [4]. Суть системы в том, что перед каждой игрой её результат предсказывается, а после завершения игры рейтинги игроков пересчитываются с учетом разницы между предсказанным счётом и

реальным. В качестве игроков рассматриваются учащийся и задача, которую он решает, в качестве их рейтингов выступают соответственно оценки знаний студента и сложности задачи. В классической шахматной системе результат игры между игроками j и k для первого из них $S_j \in \{0, 0.5, 1\}$ (проигрыш, ничья и выигрыш соответственно), а ожидаемый результат вычисляется как $E(S_j) = \frac{1}{1+10^{(\theta_j-\theta_k)/400}}$, где θ_j и θ_k – рейтинги игроков. Обновлённый рейтинг игрока j в таком случае вычисляется как $\hat{\theta}_j = \theta_j + K(S_j - E(S_j))$. Параметр K определяет то, как сильно расхождение между ожидаемым результатом и реальным влияет на изменение рейтинга. В классической модели K – константа (точнее, несколько констант для разных категорий игроков), но Марк Гликман предложил сделать это значение зависимым от конкретной ситуации[6] для большей гибкости системы.

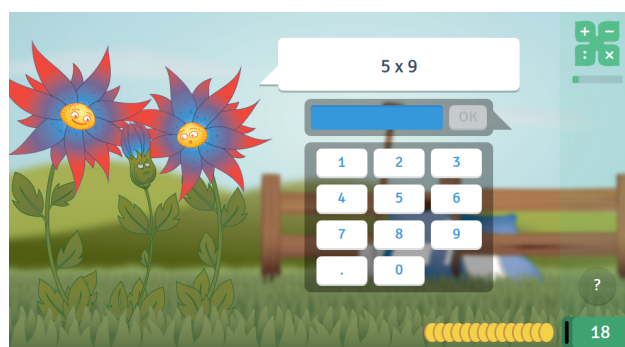
Для этого помимо сведений об уровне знаний и сложности вводятся также параметры, описывающие нашу неуверенность в текущей оценке ситуации. Это нужно для того, чтобы поведение системы варьировалось: в случае нового игрока, про которого мы почти ничего не знаем, его рейтинг мог быстро меняться с каждой игрой, а в случае профессионала, сыгравшего на нашей памяти уже сотни игр, чей рейтинг посчитан на основании большого объема данных о его уровне, один проигрыш не повлёт за собой значительного падения его рейтинга. Параметры неопределенности пересчитываются в зависимости от частоты решений и давности последнего из них. Такой параметр есть как у пользователя, так и у задачи.

4.1.3. Механика работы

На начальной странице ресурса (рис. 5а) учащемуся предлагается выбрать одну из областей для тренировки (сложение, вычитание и прочие). Области изображены в виде цветков, под каждым из которых написан текущий уровень ученика в этой области. После выбора темы пользователю предлагается несколько задач (рис. 5b), на каждую из которых отводится 20 секунд. В зависимости от правильности ответа и времени, потраченного на решение, пользователь либо получает, либо



(a) начальный экран



(b) решение задачи

Рис. 5: Интерфейс сервиса MathsGarden

лишается нескольких монет.

Пользователю предлагаются задачи, вероятность решить которые в данный момент для него ближе всего к 75%. После решения задачи вычисляется результат, зависящий от правильности ответа и от времени, потраченного на решение. Отклонение этого результата от предсказанного влияет на изменения значений сложности задачи и уровня пользователя.

4.1.4. Результаты работы

Сервис MathGarden использовался приблизительно 150 школами в течение года, после чего накопленные данные были проанализированы. Основные результаты следующие:

- полученные оценки знаний учеников коррелируют с их результатами в общем рейтинге учеников СИО. В разных темах (сложение, вычитание и прочие) корреляция составила от 0.78 до 0.84;
- надёжность оценки сложности анализировалась в числе прочего через проверку корреляции сложности зеркальных задач ($4 + 7$ и $7 + 4$ должны иметь почти одинаковую сложность), значения которой оказались между 0.88 и 0.98 среди разных тем;

- также был проведен анализ устойчивости сложности со временем. Корреляция оказалась не менее 0.95 между срезом на 44 неделе и каждой из последующих 32 недель.

4.2. Адаптивность в stepic.org

Для рекомендательной системы на stepic.org было решено добавить хендлер, который будет в рамках одного курса советовать пользователю уроки, подходящие ему в данный момент по сложности. Для вычисления сложности была использована модель, разработанная в MathsGarden. Для задач были добавлены параметры сложности (β_i) и неопределенности оценки (U_i), а для каждого пользователя и курса, который он проходит, был заведен параметр его уровня в этой теме (θ_j) и также неопределенность оценки (U_j). Формулы, приведённые ниже, взяты из описания сервиса MathsGarden[10], с небольшими изменениями для использования в нашей системе.

Для оценки результата решения задачи i пользователем j используется следующая формула: $S_{ij} = (2x_{ij} - 1)(1 - t_{ij}/d_i)$, где $x_{ij} = 1$, если ответ верный, иначе 0, t_{ij} – минимальное значение из времени, потраченного на решение, и d_i , а d_i – время, за которое в среднем пользователи решают эту задачу (в MathsGarden d_i всегда 20 секунд). Предполагаемые результаты решения вычисляются как $E(S_{ij}) = \frac{e^{2(\theta_j - \beta_i)} + 1}{e^{2(\theta_j - \beta_i)} - 1} - \frac{1}{\theta_j - \beta_i}$, где β_i – сложность задачи, θ_j – уровень пользователя[14]. Формулы учитывают время, причем чем меньше времени было потрачено, тем больше вознаграждение или штраф. Учёт времени в подсчёте результата решения позволяет без потери качества предлагать пользователю задачи с шансом решить их больше 50%, что повышает мотивацию[10][5].

Далее оценка знаний студента и сложности задачи обновляется по формулам $\hat{\theta}_j = \theta_j + K_j(S_{ij} - E(S_{ij}))$ и $\hat{\beta}_i = \beta_i + K_i(E(S_{ij}) - S_{ij})$ соответственно. Коэффициенты K_i и K_j влияют на то, как сильно различия между предсказанным исходом и реальным изменят рейтинги. Они вычисляются по формулам $K_j = K(1 + K_+U_j - K_-U_i)$,

$K_i = K(1 + K_+U_i - K_-U_j)$, где U_i и U_j – параметры студента и задачи соответственно, характеризующие нашу неуверенность в их оценках, $K = 0.0075$, $K_+ = 4$ и $K_- = 0.5$ [10]. Использование коэффициентов, зависящих от конкретных пользователей и задач, делает систему более гибкой[6].

Параметры неопределенности U зависят от частоты и давности решений, $0 \leq U \leq 1$. Пересчитываются их значения при каждом решении для пользователя и для задачи следующим образом: $\hat{U} = U - \frac{1}{40} + \frac{1}{30}D$, при каждом пересчете (использовании) неопределенность уменьшается на $\frac{1}{40}$, со временем (D – число дней без использования задачи / без практики пользователя) увеличивается. Изначально U равняется единице.

Для решения пользователю предлагаются задачи, вероятность решить которые в данный момент ближе всего к 75%.

5. Анализ результатов

Для анализа работы получившейся рекомендательной системы целесообразно выбрать интуитивно понятные метрики. Рассмотрим основные классы метрик, применяющихся для оценки рекомендательных систем[21]:

- *Предпочтение пользователей*: какую из множества систем пользователи используют наиболее охотно. Используется в случае, когда есть возможность провести сравнение, предложив различные системы нескольким группам пользователей. Метрики этого класса считаются для всех интересующих систем, после чего сравниваются.
- *Точность предсказания*: применяется в случае, если система предсказывает отношение пользователя к объектам (например, выставляемые ими рейтинги) или вероятность использования пользователем контента. К этому классу также относятся метрики, оценивающие качество ранжирования списка рекомендаций.
- *Покрытие*: какую часть из интересных пользователю объектов система смогла предсказать. Обычно затрудняется отсутствием полной информации о интересах пользователя.
- *Надежность*: можно ли в целом доверять рекомендациям. Позволяет отделить систему с в среднем неплохими рекомендациями от такой, в которой на пару отличных рекомендаций будет приходиться десяток совершенно не уместных.
- Прочие метрики, такие как доверие пользователя системе, новизна и разнообразие предлагаемых рекомендаций и прочие.

Также возможны разные режимы оценки рекомендаций:

- *офлайн-оценка*: предварительно набирается информация о пользователях, симулируется работа системы, результаты оцениваются исходя из тех же данных;
- *искусственные исследования*: специально отобранной группе пользователей предлагается опросник, который им нужно заполнить в процессе использования системы, также исследуются накопленные в процессе данные о работе системы;
- *онлайн-оценка*: рекомендательная система испытывается в «боевых условиях».

В этой работе будут использоваться оценки, посчитанные на основе реальных данных об использовании пользователями системы в нормальном режиме. Исследованы данные о простых рекомендациях пользователю на основе информации о нем, о рекомендациях в конце урока (контекстные рекомендации) и об адаптивных рекомендациях.

В первых двух случаях рекомендации показываются пользователю списком уроков, которые могут его заинтересовать. В случае простых рекомендаций в этом списке двадцать уроков, в случае контекстных – пять. Адаптивные рекомендации представляют собой один урок, который по подсчетам системы подходит пользователю в данный момент лучше всего. Будем называть каждый конкретный случай подсчета рекомендаций для пользователя *рекомендательной сессией*.

5.1. Простые рекомендации

Рекомендации для пользователя показываются списком из 20 уроков. У пользователя есть возможность узнать, почему ему был посоветован этот урок (причины берутся из хендлеров, предложивших его), а также отказаться от рекомендации конкретного урока, нажав на соответствующую кнопку возле этой рекомендации.

Для оценки будут использоваться следующие метрики:

- Процент сессий, в которых был совершен хотя бы один переход по ссылке или отказ от рекомендации. К сожалению, так как

рекомендации показываются на главной странице, где не многие их действительно видят, этот процент будет невелик. Дальнейшие метрики будут рассматриваться только для сессий, на которые была получена какая-то реакция, положительная (переход по ссылке) или отрицательная (отказ от рекомендации);

- Среднее число переходов по ссылке в сессии;
- Какую часть урока в среднем проходят пользователи после перехода по ссылке;
- Сколько совершено отказов от рекомендаций.

Были исследованы данные о работе системы за четыре месяца. Результаты представлены в таблице 2.

Таблица 2: Результаты для простых рекомендаций

Метрика	Значение
Число сессий	381868
Число сессий с реакцией	18066
Процент сессий с реакцией	4.7%
Число открытых рекомендаций (из 20)	1.6
Пройденная часть урока	0.52
Число отказов от рекомендации	184

На рис. 6 представлены распределения числа открытых рекомендаций и пройденной части урока.

5.2. Контекстные рекомендации

В случае, если урок, пройденный пользователем, не включен ни в какой курс, то есть не имеет явного следующего за ним урока, пользователю предлагаются рекомендации, сделанные на основе информации о нем и о текущем уроке. В данном случае пользователь наверняка видел рекомендации, что прямо отражается на проценте

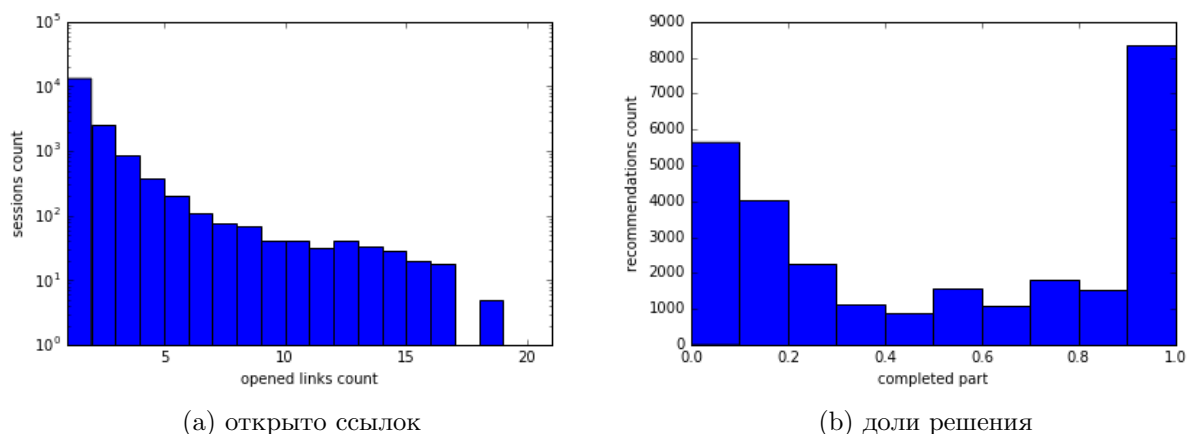


Рис. 6: Распределения для простых рекомендаций

рекомендательных сессий с реакцией.

В таблице 3 также отражены результаты за четыре месяца.

Таблица 3: Результаты для контекстных рекомендаций

Метрика	Значение
Число сессий	26995
Число сессий с реакцией	15125
Процент сессий с реакцией	56%
Число открытых рекомендаций (из 5)	1.48
Пройденная часть урока	0.5
Число отказов от рекомендации	0

На рис. 7 представлены распределения числа открытых рекомендаций и пройденной части урока.

5.3. Адаптивные рекомендации

Для оценки качества адаптивных рекомендаций использовались данные, накопленные за две недели работы. Эти две недели пользователям было предложено проходить в адаптивном режиме курс-тренажер по языку программирования Python.

Пользователь получал рекомендованный урок, после чего он мог пометить его слишком сложным, слишком простым или решить его.

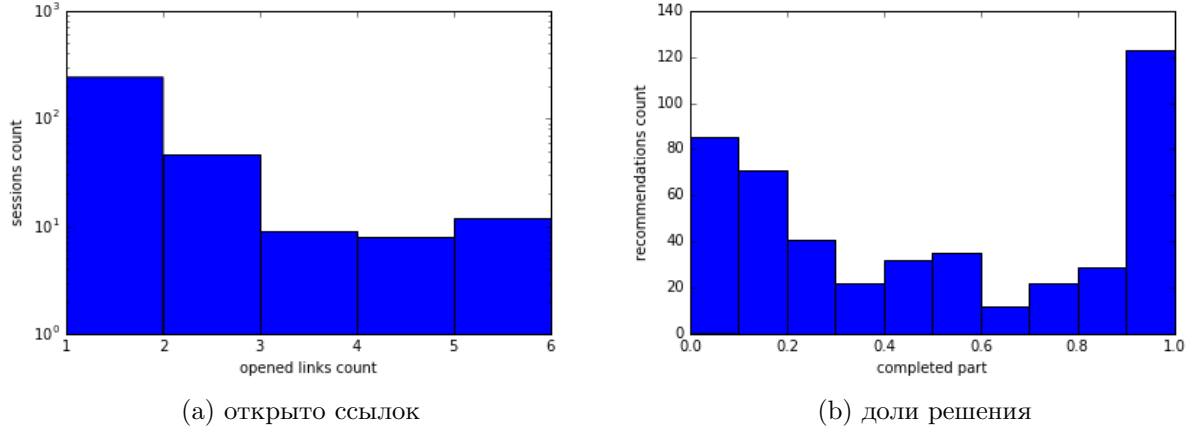


Рис. 7: Распределения для контекстных рекомендаций

Его действия влияли на то, какой уровень знаний (*skill*) система присваивала ему, и какую сложность (*difficulty*) – задаче. При этом для каждой рекомендации рассчитывалось предсказание исхода (1 – решил, 0 – не решил). Это позволяет вычислить средне-квадратичную ошибку (mean square error, MSE) предсказания:

$$MSE = \sum_{k=0}^N (predicted_k - real_k)^2,$$

где N – число примеров, $predicted_k$ и $real_k$ – предсказанный и реальный результаты для k -го примера.

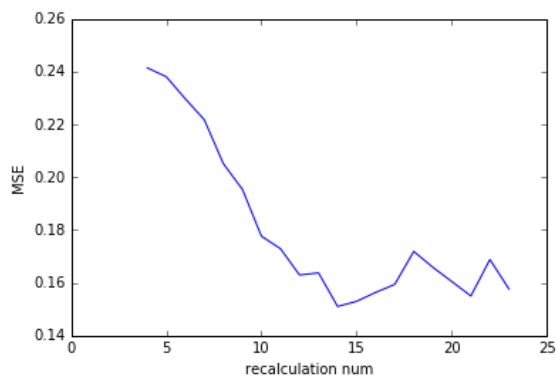
В таблице 4 представлены основные метрики работы системы.

Таблица 4: Результаты для адаптивных рекомендаций

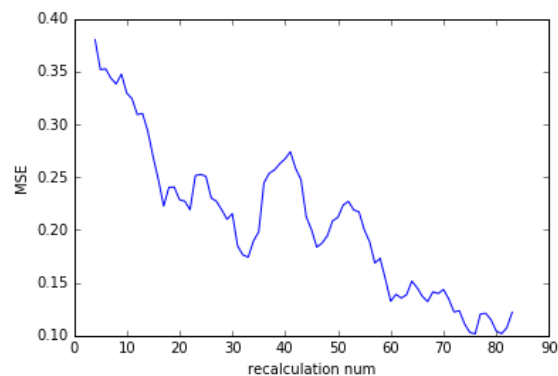
Метрика	Значение
Число сессий	1511
Число пользователей	246
Реакций «решено»	1329
Реакций «слишком просто»	95
Реакций «слишком сложно»	87
MSE предсказания исхода	0.22

Также было исследовано, действительно ли ошибка предсказания

для конкретного пользователя или задачи уменьшается со временем. Для этого ошибки были усреднены по пользователям (рис. 8a) и по задачам (рис. 8b). На получившихся графиках наблюдается явное, хоть и не равномерное, убывание ошибки.



(a) пересчет знаний пользователя



(b) пересчет сложности задачи

Рис. 8: Изменение ошибки предсказания при пересчете

6. Заключение

В рамках работы была реализована рекомендательная система для образовательного контента, которая теперь используется на платформе *stepic.org*. Эта система совмещает в себе фильтрацию контента и коллаборативную фильтрацию, а также имеет возможность прохождения материала в адаптивном режиме, по индивидуальному для каждого учащегося пути.

Структура системы позволяет дополнять ее новыми способами рекомендации, причем значимость каждого способа и его вклад в итоговые рекомендации будет определяться системой автоматически на основе реакции пользователей.

Также система предоставляет возможность получать рекомендации в разных ситуациях: как просто при входе на сайт, на домашней странице, так и в процессе обучения, сразу после прохождения урока. Помимо этого имеются также адаптивные рекомендации, подстраивающиеся под уровень знаний конкретного пользователя.

В качестве дальнейших путей развития системы можно указать, во-первых, создание новых правил фильтрации контента исходя из нужд пользователей, во-вторых, совершенствование рекомендательной системы: используя ручную разметку контента можно при рекомендациях использовать изученные и не изученные пользователем темы, предлагать пройти ему что-то, знаний о чем ему не хватает для изучения интересного ему материала. Также возможна работа над проблемой «холодного старта», ситуации, когда про пользователя и про материалы нет достаточного объема накопленных данных, что мешает с уверенностью давать рекомендации.

Список литературы

- [1] Anand Sarabjot Singh, Mobasher Bamshad. Intelligent Techniques for Web Personalization // Proceedings of the 2003 International Conference on Intelligent Techniques for Web Personalization. — ITWP'03. — Berlin, Heidelberg : Springer-Verlag, 2005. — P. 1–36.
- [2] Coursera. — <https://www.coursera.org/about/>.
- [3] Creative Commons. — <http://creativecommons.org/>.
- [4] Elo A.E. The rating of chessplayers, past and present. — Arco Pub., 1978.
- [5] Ericsson K Anders et al. The influence of experience and deliberate practice on the development of superior expert performance.
- [6] Glickman Mark E. A comprehensive guide to chess ratings // American Chess Journal. — 1995. — Vol. 3. — P. 59–102.
- [7] Harasim Linda Marie. Learning networks: A field guide to teaching and learning online. — MIT press, 1995.
- [8] Hastie T., Tibshirani R., Friedman J.H. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer series in statistics. — Springer, 2001.
- [9] IMDb. — <http://www.imdb.com/>.
- [10] Klinkenberg S, Straatemeier M, Van der Maas HLJ. Computer adaptive practice of maths ability using a new item response model for on the fly ability and difficulty estimation // Computers & Education. — 2011. — Vol. 57, no. 2. — P. 1813–1824.
- [11] Knewton Adaptive Learning. — <https://www.knewton.com/wp-content/uploads/knewton-adaptive-learning-whitepaper.pdf>.

- [12] Lord F.M. Applications of Item Response Theory to Practical Testing Problems. — Erlbaum Associates, 1980.
- [13] Mahmood Tariq, Ricci Francesco. Improving Recommender Systems with Adaptive Conversational Strategies // Proceedings of the 20th ACM Conference on Hypertext and Hypermedia. — HT '09. — New York, NY, USA : ACM, 2009. — P. 73–82.
- [14] Maris Gunter, Maas Han. Speed-Accuracy Response Models: Scoring Rules based on Response Time and Accuracy // Psychometrika. — 2012. — Vol. 77, no. 4. — P. 615–633.
- [15] MovieLens. — <https://movielens.org/>.
- [16] Netflix. — <https://www.netflix.com>.
- [17] Ng Andrew Y. Feature selection, l1 vs. l2 regularization, and rotational invariance // In ICML. — 2004.
- [18] Rasch G. Probabilistic Models for Some Intelligence and Attainment Tests. / Ed. by M. E. S. A. Press. — MESA Press, 1960.
- [19] Recommender Systems Handbook / Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kantor. — 1st edition. — New York, NY, USA : Springer-Verlag New York, Inc., 2010.
- [20] Jones Eric, Oliphant Travis, Peterson Pearu et al. SciPy: Open source scientific tools for Python. — 2001–. — [Online; accessed 2015-12-06]. URL: <http://www.scipy.org/>.
- [21] Shani Guy, Gunawardana Asela. Evaluating Recommendation Systems // Recommender Systems Handbook. — Springer, 2011.
- [22] Stepic, an educational engine. — <https://stepic.org>.
- [23] Tang T., McCalla G. Smart Recommendation for an Evolving e-Learning System: Architecture and Experiment // International Journal on e-Learning. — 2005. — Vol. 4, no. 1. — P. 105–129.

- [24] Tibshirani Robert. Regression Shrinkage and Selection Via the Lasso // Journal of the Royal Statistical Society, Series B. — 1994. — Vol. 58. — P. 267–288.
- [25] Tikhonov A. N. Solution of incorrectly formulated problems and the regularization method // Soviet Math. Dokl. — 1963. — Vol. 4. — P. 1035–1038.
- [26] Udacity. — <https://www.udacity.com/>.
- [27] Using collaborative filtering to weave an information tapestry / David Goldberg, David Nichols, Brian M Oki, Douglas Terry // Communications of the ACM. — 1992. — Vol. 35, no. 12. — P. 61–70.
- [28] Wikidata. — <https://www.wikidata.org>.
- [29] edX. — <https://www.edx.org>.
- [30] A sneak preview to the chapter “Recommender Systems in Technology Enhanced Learning” / Nikos Manouselis, Hendrik Drachsler, Riina Vuorikari et al. // Recommender Systems Handbook. — Springer, 2011.
- [31] van der Linden W. J., Glas C. A. W. Computerized adaptive testing: Theory and practice / Kluwer Academic Publishers. — Dordrecht, The Netherlands : Kluwer Academic Publishers, 2000.